Constructing Sailing Match Race Schedules: Round-Robin Pairing Lists

Craig Macdonald, Ciaran McCreesh, Alice Miller, and Patrick Prosser

University of Glasgow, Glasgow, Scotland
patrick.prosser@glasgow.ac.uk

Abstract. We present a constraint programming solution to the problem of generating round-robin schedules for sailing match races. Our schedules satisfy the criteria published by the International Sailing Federation (ISAF) governing body for match race pairing lists. We show that some published ISAF schedules are in fact illegal, and present corresponding legal instances and schedules that have not previously been published. Our schedules can be downloaded as blanks, then populated with actual competitors and used in match racing competitions.

1 Introduction

This work describes a round-robin competition format that arises from sailing, known as match racing. Competitors, i.e. *skippers*, compete against each other in a series of matches taking place in rounds, known as *flights*. A match is composed of two skippers, with each skipper in a boat on their own. Skippers in a match set off together, one on the port side, the other starboard, and first home wins that match. Criteria published by the International Sailing Federation (ISAF) dictate what makes a legal match race schedule and what should be optimized. This is a rich set of criteria, 13 in all, and as far as we know, all published schedules have been produced by hand. This is a daunting task. A close inspection of these schedules reveals that most are illegal, violating many of the match race scheduling criteria, many schedules are missing, and some that are legal are far from the ISAF definition of optimality.

This paper describes the scheduling problem. We present a constraint programming solution and document how this model was constructed incrementally. We believe this illustrates how adaptable constraint programming is: a robust model can be developed incrementally to address all of the required criteria. Our schedules are then constructed in a sequence of stages, not dissimilar to the hybrid and decomposition based approaches by Lombardi and Milano [3]. Some ISAF published schedules are presented along with some of our own, some of these being new. These schedules are essentially blank forms, made available to event organizers, which can be filled in with names of actual competitors, i.e. these schedules are reusable.

The paper is organized as follows: we introduce the 13 match racing criteria, then explain the constraint models and four stages of schedule construction, and then schedules are presented. We finish with discussion and a conclusion.

2 Problem Definition: Round-Robin Pairing Lists

The guidelines for running match racing events [2] places a number of criteria on how the competing skippers are scheduled into flights and matches, known as pairing lists. Several of these criteria are applicable when the number of skippers exceeds the number of available boats, in which case skippers have to change in and out of boats, and are allowed time in the schedule to check and fine-tune the boat they change into. Typically, matches set off at 5 minute intervals with each match taking about 20 minutes. Therefore, if we have 10 skippers and 10 boats we have 45 matches, 9 flights each of 5 matches, and if each flight takes about 50 minutes, eight or nine flights a day is a reasonable achievement for most events [2]. Consequently, the number of boats and skippers involved is typically small.

The criteria for match racing schedules (pairing lists) are given in ISAF "International Umpires' and Match Racing Manual" [2], section M.2 "Recommended Criteria for Round Robin Pairing Lists", and are detailed below.

Principal Criteria in Order of Priority:

1. Each skipper sails against each other skipper once.
2*. When skippers have an even number of matches, they have the same number of port and starboard assignments.
3*. When skippers have an odd number of matches, the first half of the skippers will have one more starboard assignment.
4. No skipper in the last match of a flight should be in the first match of the next flight.
5. No skipper should have more than two consecutive port or starboard assignments.
6. Each skipper should be assigned to match 1, match 2, etc. in a flight as equally as possible.
7. In flights with five or more matches, no skipper should be in the next-to-last match in a flight and then in the first match of the next flight.
8. If possible, a skipper should be starboard when meeting the nearest lowest ranked skipper (i.e. #1 will be starboard against #2, #3 will be starboard against #4).
9. Close-ranked skippers meet in the last flight.
10. Minimize the number of boat changes.
11. Skippers in the last match of a flight do not change boats.
12. Skippers in new boats do not sail in the first match of the next flight.
13. Skippers have a reasonable sequence of matches and blanks.

Note that criteria 10, 11 and 12 only apply when there are fewer boats than skippers; 11 and 12 override 6 when changes are required, and 13 applies when there are more boats than skippers.

We have rephrased criteria 2 and 3 (denoted *) to clarify an error in the manual [2]. Note that the order of matches within a flight is significant and is constrained, as is the order of flights within a schedule and the position of skippers within a match. This permits fair schedules that provide sufficient changeover time for skippers changing boats, etc.

Criterion 6 allows us to measure the *balance* of a schedule. Perfect balance would be when a skipper has as many matches in first position in flights as in second position in flights, and so on. For example, if we had 9 skippers, each skipper would have 8 matches and could be in the first, second, third or fourth match of a flight. In perfect balance, each of the 8 skippers would appear twice in each position. Balance is one of our optimization criteria, and we do this unconditionally, even when there are more skippers than boats.

Criterion 13 uses the term *blanks* where conventionally we use the term *bye*. That is, a blank is a bye and a bye is a flight where a skipper does not compete (and is ashore).

Criterion 12 discusses boat changes: if in flight i this skipper is not in a match (i.e. it is a bye for this skipper) but is in a match in flight $i + 1$ then he has to get into a boat, rearrange all his kit and set the boat to his preference before racing commences, and this is a boat change.

Next, we note that skippers can be ranked prior to an event, based on their performance in previous events[1], which is used to seed the skippers. The ordering of skippers within a match signifies their starting position (port or starboard), as skippers allocated the starboard starting position gain a small competitive advantage in that match—criterion 8 accomplishes the seedings.

Criterion 10 is ambiguous: this could be interpreted on a schedule basis: i.e. to minimize the overall number of changes in the entire pairing list schedule; or alternatively as well as a fair, but minimal number of changes across all skippers. In our work, we minimize the overall number of changes.

There are some conflicts inherent in the criteria. Take criteria 4 and 11, assume we have 4 boats, and in flight i the last match is the pair of skippers (x, y). Criterion 11 dictates that skippers x and y must appear in the next flight, $i + 1$, and criterion 4 that neither can be first in the next flight. This forces skippers x and y to compete in flight $i + 1$ as the last match, violating criterion 1. Therefore, although not explicitly stated, it is not possible to satisfy every criteria with fewer than 6 boats.

3 The Constraint Models

Our approach produces schedules in four stages. The first stage produces a schedule that respects criteria 1, 4, 11 and 12, and minimizes criteria 10 (boat changes). The second stage constructs a schedule that respects criteria 1, 4, 11, 12 (again), has criterion 10 as a hard constraint generated from first stage, and minimizes criterion 6 (balance). This results in a multi-objective schedule that

[1] `http://www.sailing.org/rankings/match/`

Flight	Matches		
0	(0,1)	(2,3)	(4,5)
1	(0,2)	(4,6)	(1,5)
2	(2,6)	(0,5)	(1,3)
3	(5,6)	(0,3)	(1,4)
4	(3,5)	(1,6)	(2,4)
5	(3,6)	(0,4)	(2,5)
6	(3,4)	(1,2)	(0,6)

Table 1. A round-robin schedule with 7 skippers, 6 boats and 7 flights. Skippers are numbered 0 to 6. Note that the order of skippers within a flight is significant, as is position within a match (port or starboard). This schedule is the result of stages 1 and 2, and has yet to be renumbered and oriented (stages 3 and 4).

attempts to minimize boat changes *and* balance. The third stage satisfies criterion 9, and is a simple translation of the schedule produced in stage 2. The final stage orients skippers within matches to address criteria 2, 3, 5 and 8.

We now describe the constraint models and processes used in each stage. In the text below we assume there are n skippers and b boats, with $m = b/2$ matches in a flight. In total there are $t = n(n-1)/2$ matches and $f = \lceil n(n-1)/m \rceil$ flights. We use the schedule in Table 1, for 7 skippers and 6 boats, to help illustrate the constraint models.

3.1 Stage 1: Minimizing Boat Changes

Modeling skippers: The first thing we model, in the first half of Figure 1, is a skipper. Each skipper σ has both a temporal view of their schedule ("who, if anyone, am I racing in the match at time t?"), and a state view ("where am I racing within flight f?"). The temporal view is via the array *timeSlot* defined in (V1). If variable $timeSlot[i] = k$ and $k \geq 0$ then this skipper is in a match with skipper k at time i.

Variable $state[i]$ (V2) gives the state of this skipper in flight i, corresponding to time slots $timeSlot[m \cdot i]$ to $timeSlot[m(i+1)-1]$. The cases in (C1) state that a skipper can be in a match in the first time slot in the flight, or in the middle of the flight[2], or in the last time slot of the flight. Alternatively, if all time slots in a flight are equal to -1 then the skipper is in a bye (i.e. not in a match in this flight), and if all time slots in a flight are equal to -2 then the skipper has finished all matches.

We must then ensure that each skipper σ is in a match with all other skippers $\{0, \ldots, n-1\} \setminus \{\sigma\}$. This is (C2), which is enforced by imposing a global cardinality constraint [5] on the array *timeSlot*.

State transitions: The *state* variables are then used to impose match race criterion 4 (if last in flight i then not first in flight $i+1$), criterion 11 (if last in flight i then not in a bye in flight $i+1$) and criterion 12 (if flight i is a bye then flight $i+1$ is not first). These criteria are imposed in (C3) by the deterministic finite automaton (DFA) shown in Figure 2 using Pesant's *regular* constraint [4].

[2] i.e. not first and not last.

A copy of these variables and constraints is created for each skipper σ:

$\forall \tau \in \{0 \mathinner{..} t{-}1\} : \mathit{timeSlot}[\tau] \in \{-2 \mathinner{..} t{-}1\} \setminus \{\sigma\}$ (V1)

$\forall i \in \{0 \mathinner{..} f{-}1\} : \mathit{state}[i] \in \{\textsc{First}, \textsc{Mid}, \textsc{Last}, \textsc{Bye}, \textsc{End}\}$ (V2)

$\forall i \in \{0 \mathinner{..} f{-}1\} :$ (C1)

$\mathit{state}[i] = \textsc{First} \Leftrightarrow \mathit{timeSlot}[m \cdot i] \geq 0$

$\mathit{state}[i] = \textsc{Mid} \Leftrightarrow \exists j \in \{m \cdot i + 1 \mathinner{..} m \cdot (i+1) - 2\} : \mathit{timeSlot}[j] \geq 0$

$\mathit{state}[i] = \textsc{Last} \Leftrightarrow \mathit{timeSlot}[m \cdot (i+1) - 1] \geq 0$

$\mathit{state}[i] = \textsc{Bye} \Leftrightarrow \forall j \in \{m \cdot i \mathinner{..} m \cdot (i+1) - 1\} : \mathit{timeSlot}[j] = -1$

$\mathit{state}[i] = \textsc{End} \Leftrightarrow \forall j \in \{m \cdot i \mathinner{..} m \cdot (i+1) - 1\} : \mathit{timeSlot}[j] = -2$

$\text{eachOccursExactlyOnce}(\mathit{timeSlot}, \{0, \ldots, n{-}1\} \setminus \{\sigma\})$ (C2)

$\text{regular}(\mathit{state}, \text{Figure 2})$ (C3)

$\forall i \in \{0 \mathinner{..} f{-}2\} : \mathit{change}[i] \in \{0, 1\}$ (V3)

$\mathit{totalChanges} \in \mathbb{N}$ (V4)

$\forall i \in \{0 \mathinner{..} f{-}2\} : \mathit{change}[i] = 1 \Leftrightarrow \mathit{state}[i] = \textsc{Bye} \wedge \mathit{state}[i+1] \neq \textsc{Bye}$ (C4)

$\mathit{totalChanges} = \sum \mathit{change}$ (C5)

Match and temporal perspectives:

$\forall i \in \{0 \mathinner{..} n{-}2\} : \forall j \in \{i{+}1 \mathinner{..} n{-}1\} :$

$\mathit{match}[i, j] \in \{0 \mathinner{..} t{-}1\}$ (V5)

$\mathit{match}[j, i] \equiv \mathit{match}[i, j]$

$\forall k \in \{0 \mathinner{..} t{-}1\} :$

$\mathit{match}[i, j] = k \Leftrightarrow \sigma[i].\mathit{timeSlot}[k] = j \wedge \sigma[j].\mathit{timeSlot}[k] = i$ (C6)

$\forall i \in \{0 \mathinner{..} n{-}2\} : \forall j \in \{i{+}1 \mathinner{..} n{-}1\} :$

$\mathit{modMatch}[i, j] \in \{0 \mathinner{..} f{-}1\}$ (V6)

$\mathit{modMatch}[j, i] \equiv \mathit{modMatch}[i, j]$

$\mathit{modMatch}[i, j] = \mathit{match}[i, j] / m$ (C7)

$\forall i \in \{0 \mathinner{..} n{-}1\} : \text{allDifferent}(\mathit{modMatch}[i])$ (C8)

$\forall \tau \in \{0 \mathinner{..} t{-}1\} :$

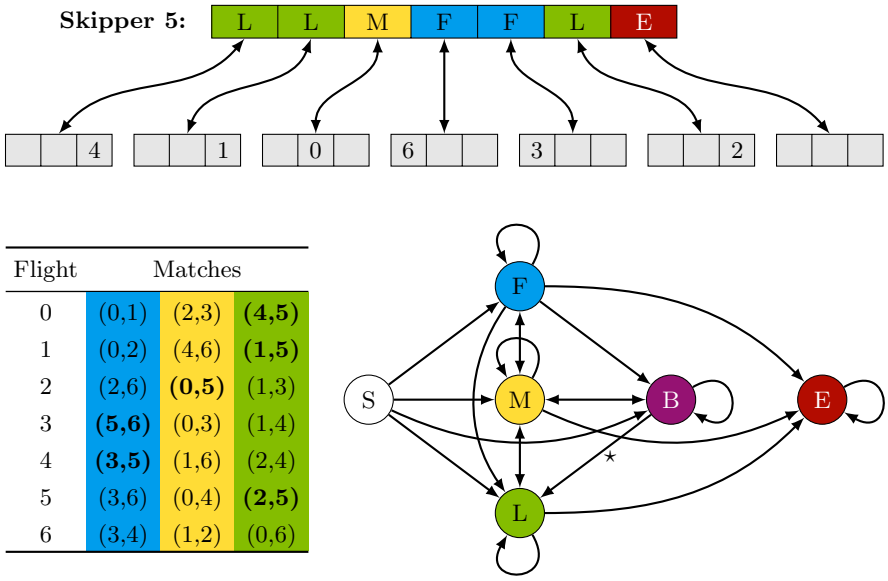
$\mathit{time}[\tau] \in \{(0, 1) \mathinner{..} (n{-}2, n{-}1)\}$ (V7)

$\forall i \in \{0 \mathinner{..} n{-}2\} : \forall j \in \{i{+}1 \mathinner{..} n{-}1\} : \mathit{time}[\tau] = (i, j) \Leftrightarrow \mathit{match}[i, j] = \tau$ (C9)

$\mathit{totalBoatChanges} = \sum \sigma.\mathit{totalChanges}$ (V8)

$\text{minimise}(\mathit{totalBoatChanges})$ (C10)

Fig. 1. Our constraint model, from a skipper perspective (top) and a match and temporal perspective (below). The number of skippers is n, and m is the number of matches in a flight. The number of flights is $f = \lceil n(n-1)/m \rceil$, and there are $t = n(n-1)/2$ matches (and time slots) in total. We define $\mathbb{N}$ to include zero.



Flight	Matches		
0	(0,1)	(2,3)	**(4,5)**
1	(0,2)	(4,6)	**(1,5)**
2	(2,6)	**(0,5)**	(1,3)
3	**(5,6)**	(0,3)	(1,4)
4	**(3,5)**	(1,6)	(2,4)
5	(3,6)	(0,4)	**(2,5)**
6	(3,4)	(1,2)	(0,6)

Fig. 2. A pictorial representation of a skipper (skipper 5) with multicoloured state and grey timeslots. The schedule for skipper 5 is in bold and the DFA for criteria 4, 11 and 12 is drawn with state START in white, FIRST in blue, MID in yellow, LAST in green, BYE in pink and END in red. The edge marked $\star$ is explained in the text.

The arcs in Figure 2 represent possible transitions from one state to another. The DFA is encoded as a set of transition objects $\langle q_i, \iota, q_j \rangle$, where a transition is made from state q_i to state q_j when encountering input ι. In addition we specify the accepting states, which are all states except the start state (white) and the bye state (pink). This constraint also ensures that if a skipper has finished all matches in flight i (i.e. is in state END) then that skipper will also have finished all matches in flight $i + 1$ (and so on). Note that when we have as many boats as skippers, the directed edge (BYE, LAST) (marked $\star$) becomes bidirectional.

Figure 2 also shows the skipper-oriented variables and constraints corresponding to skipper 5 in the schedule shown in Table 1. The *state* array is presented as coloured boxes, and below that is the *timeSlot* array. The arrows represent the channeling constraints (C1). The *state* array is coloured with green representing LAST, yellow for MID, blue for FIRST, red for LAST and (not shown) pink for BYE. The schedule of Table 1 is reproduced with the matches for skipper 5 emboldened.

Boat changes: A boat change occurs when a skipper has been in a BYE state in flight i and then is in a match in the next flight $i+1$. This is encoded using the array of zero/one variables (V3), and the constraint (C4), and accumulated into *totalChanges* (V4, C5).

Match perspective: We are now in a position to model criterion 1 (each skipper sails against every other skipper) and optimization criterion 10 (minimize boat changes). In the second half of Figure 1 we present a match perspective of the schedule, using a two dimensional array of variables *match* (V5), where $match[i, j]$ is the time slot in which skippers $\sigma[i]$ and $\sigma[j]$ meet in a match. Only the half above the diagonal is represented, and the lower half is made up of exactly the same variables, i.e. $match[i, j]$ is *exactly* the same constrained variable as $match[j, i]$. Constraint (C6) states that a match between skippers $\sigma[i]$ and $\sigma[j]$ takes place at time k (i.e. $match[i, j] = k$) if and only if skipper $\sigma[i]$'s k^{th} time slot is skipper $\sigma[j]$ and conversely that skipper $\sigma[j]$'s k^{th} time slot is skipper $\sigma[i]$. Variables (V6) and constraints (C7) then convert this from time slots to flights, i.e. $modMatch[i, j]$ is the flight in which skippers $\sigma[i]$ and $\sigma[j]$ meet for a match. Finally, constraint (C8) ensures that each skipper's match occurs in different flights. Also, since $modMatch[i, j] \equiv modMatch[j, i]$, setting rows to be all different also ensures that columns are all different.

Temporal perspective: We now take a temporal view (V7), such that $time[\tau]$ is a pair (i, j), stating that at time τ skippers $\sigma[i]$ and $\sigma[j]$ are in a match. We channel between the match perspective and the temporal perspective using (C9).

Optimization criteria: Finally we have the optimization criterion (criterion 10) to minimize the number of boat changes. This is handled by (V8) and (C10).

Decision variables: The decision variables are $time[0]$ to $time[t-1]$, i.e. for each time slot we decide what match takes place. A symmetry break is made at the top of search by forcing the first round to contain the matches $(0, 1), (2, 3), \ldots$, i.e. $\forall i \in \{0 \ldots m-1\} : match[2i, 2i+1] = i$. (This is independent of criterion 9, which will be enforced in a subsequent stage by renumbering.)

Figure 3 gives a pictorial view of the entire model, using the 7 skipper and 6 boat problem. On the right we have the 7 skippers with their states and time slots. Again, on the right we give the schedule actually produced. On the left we have the *modMatch* and *match* arrays, and at the bottom the *time* array. The arrows show the channeling between parts of the model.

The schedule presented has 6 boat changes: there are no boat changes in flight 0 (first flight), in flight 1 skipper 6 has a boat change, in flight 2 skipper 3 has a boat change, in flight 3 skipper 4, in flight 4 skipper 6, in flight 5 skipper 0, and flight 6 skipper 1.

3.2 Stage 2: Minimizing Imbalances

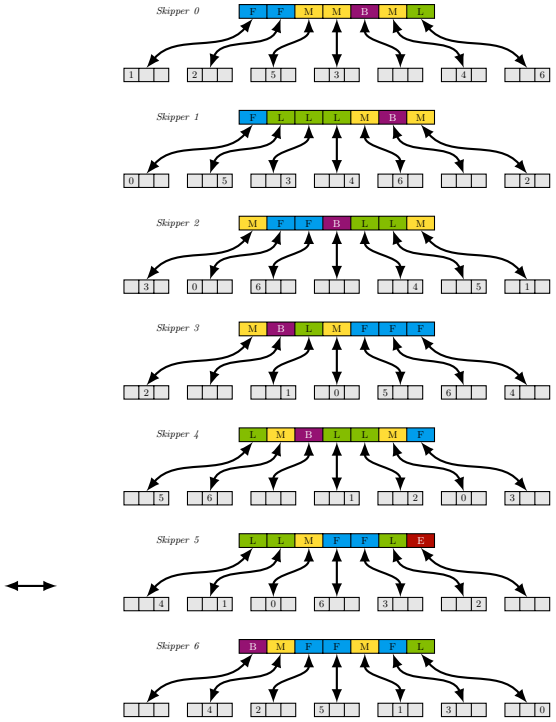
Having produced a schedule that minimizes boat changes, we then minimize imbalance (i.e. apply criterion 6) by extending the model in Figure 4. Assuming the first stage produces a schedule with β boat changes we post this as a hard constraint (C11) in stage 2. For each skipper σ we introduce a zero/one variable $position[i, j]$ (V9) which is 1 if and only if skipper σ is in a match in time $m \cdot j + i$,

modMatch

	0	1	2	3	4	5	6
0	-	0	1	3	5	2	6
1	0	-	6	2	3	1	4
2	1	6	-	0	4	5	2
3	3	2	0	-	6	4	5
4	5	3	4	6	-	0	1
5	2	1	5	4	0	-	3
6	6	4	2	5	1	3	-

match

	0	1	2	3	4	5	6
0	-	0	3	10	16	7	20
1		-	19	8	11	5	13
2			-	1	14	17	6
3				-	18	12	15
4					-	2	4
5						-	9
6							-



schedule

Flight	Matches		
0	(0,1)	(2,3)	(4,5)
1	(0,2)	(4,6)	(1,5)
2	(2,6)	(0,5)	(1,3)
3	(5,6)	(0,3)	(1,4)
4	(3,5)	(1,6)	(2,4)
5	(3,6)	(0,4)	(2,5)
6	(3,4)	(1,2)	(0,6)

time

(0,1)	(2,3)	(4,5)	(0,2)	(4,6)	(1,5)	(2,6)	(0,5)	(1,3)	(5,6)	(0,3)	(1,4)	(3,5)	(1,6)	(2,4)	(3,6)	(0,4)	(2,5)	(3,4)	(1,2)	(0,6)

Fig. 3. A pictorial representation of the entire model of the schedule for 7 skippers and 6 boats. The schedule is reproduced on the right. In the centre we have the 7 skippers, on left the *modMatch* and *match* arrays, and along the bottom the *time* array. The arrows signify channeling between parts of the model.

The objective value from stage 1 is used as a hard constraint:

$$totalBoatChanges \leq \beta \tag{C11}$$

A copy of these variables and constraints is created for each skipper σ:

$$\forall i \in \{0 \dots m{-}1\} : \forall j \in \{0 \dots f{-}1\} :$$

$$position[i,j] \in \{0,1\} \tag{V9}$$

$$position[i,j] = 1 \Leftrightarrow timeSlot[m.j+i] \geq 0 \tag{C12}$$

$$\forall i \in \{0 \dots m{-}1\} :$$

$$imbalance[i] \in \mathbb{N} \tag{V10}$$

$$imbalance[i] = \left| \frac{n-1}{m} - \sum_{j=0}^{f-1} position[i,j] \right| \tag{C13}$$

$$maxImbalance \in \mathbb{N} \tag{V11}$$

$$maxImbalance = \text{maximum}(imbalance) \tag{C14}$$

We minimize the maximum imbalance over all skippers:

$$\forall i \in \{0 \dots n{-}1\} : imbalance[i] \equiv \sigma[i].maxImbalance \tag{V12}$$

$$maxImbalance \in \mathbb{N} \tag{V13}$$

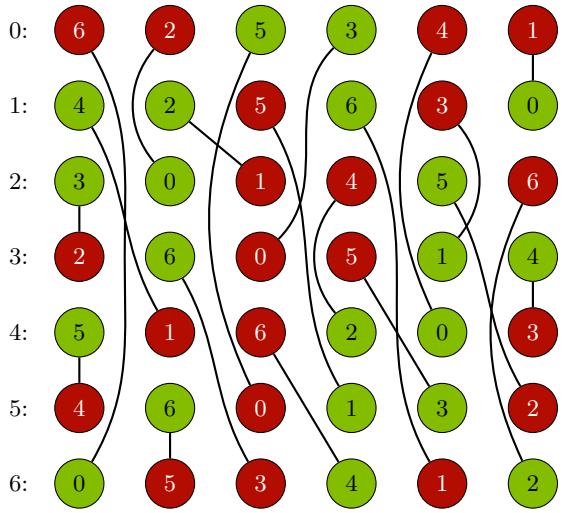
$$maxImbalance = \text{maximum}(imbalance) \tag{C15}$$

$$\text{minimise}(maxImbalance) \tag{C16}$$

Fig. 4. Additions to the constraint model in Figure 1 for stage 2. The constant β is the minimum number of boat changes found in stage 1.

Flight	Matches		
0	(0,6)	(2,3)	(4,5)
1	(0,2)	(4,1)	(6,5)
2	(2,1)	(0,5)	(6,3)
3	(5,1)	(0,3)	(6,4)
4	(3,5)	(6,1)	(2,4)
5	(3,1)	(0,4)	(2,5)
6	(3,4)	(6,2)	(0,1)

Table 2. A round-robin schedule with 7 skippers, 6 boats and 7 flights, which is the result of stages 1, 2, and 3. It has 6 boat changes and imbalance 1. An example of imbalance is skipper $\sigma[0]$ with 2 matches in position 0 (first), 3 in position 1 and 1 (mid) in position 2 (last). Perfect balance would be 2 matches in each position, only achieved by $\sigma[2]$.



Flight	Matches		
0	(0,6)	(3,2)	(5,4)
1	(0,2)	(4,1)	(6,5)
2	(2,1)	(5,0)	(6,3)
3	(1,5)	(3,0)	(4,6)
4	(3,5)	(6,1)	(2,4)
5	(1,3)	(0,4)	(5,2)
6	(4,3)	(2,6)	(1,0)

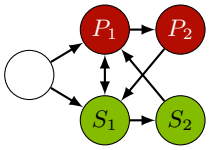


Fig. 5. A pictorial representation of the orientation process, stage 4. Top right is the actual oriented schedule. Below that is the DFA to satisfy criterion 5. On the left is the zero/one variables, a row representing a skipper's sequence of competitors. An edge between variables corresponds to a matched pair, that must take different values.

i.e. in the i^{th} match of the j^{th} flight (C12). Imbalance (V10) is then computed for each position, as the absolute amount over or under the expected presence in a given position (C13). Finally, we compute the maximum imbalance over all positions for this skipper (V11, C14).

We then capture the maximum imbalance over all skippers (V12, V13 and C15) and minimize this (C16). As before, the decision variables are $time[0]$ to $time[t-1]$ and the same symmetry breaking is imposed at top of search.

3.3 Stage 3: Renaming Skippers

Stage 3 then takes as input the schedule produced in stage 2 and renames skippers in the last round of the last flight to be $(0, 1)$, satisfying criterion 9. This is a simple linear process. The schedule produced for 7 skippers and 6 boats is shown in Table 2.

3.4 Stage 4: Orienting Matches

The final stage, stage 4, orients skippers within matches, such that they are either port (red) or starboard (green). This is done to satisfy criteria 2, 3, 5 and 8. A sketch of the constraint model is given with a supporting diagram, Figure 5. A two dimensional array of constrained integer variables $orient[i, j] \in \{0, 1\}$ has $orient[i, j] = 0$ if and only if skipper $\sigma[i]$ is on port side in his j^{th} match (starboard otherwise). For example, taking the schedule in Table 2, skipper

$\sigma[2]$ meets skippers in the order 3,0,1,4,5,6; skipper $\sigma[3]$ meets skippers in the order 2,6,0,5,1,4 and $\sigma[5]$ meets skippers in the order 4,6,0,1,3,2. Therefore $orient[2,0] = 1$ *and* $orient[3,0] = 0$ (criterion 8), $orient[2,4] \neq orient[5,5]$ and $orient[3,3] \neq orient[5,4]$. Summation constraints on each skipper enforce criteria 2 and 3 (equal number of port and starboard matches). Criterion 8 is enforced by posting constraints such that in a match $(\sigma[i], \sigma[j])$ where $|i-j| = 1$, the higher indexed skipper (lower ranked) is on the port side. A *regular* constraint is posted for criterion 5 (restricting sequences of port and starboard assignments).

In Figure 5, port is red and starboard green. The sequence of zero/one variables for each skipper is shown on the left. Bottom right is the DFA, and top right is the final schedule produced from stages 1 to 4. The schedule has 6 boat changes and a maximum imbalance of 1. The schedule presented here is new, not appearing in the example schedules provided in the manual [2].

4 Sample Schedules

We present four pairs of schedules. The first pair is for 8 skippers and 6 boats, one published by ISAF, the other generated by us. Next, we present two new schedules, not appearing in the ISAF manual, for 9 skippers and 6 boats and 9 skippers with 8 boats, then 10 skippers and 6 boats, again both published by ISAF and generated by us. Finally, we show what is a relatively easy problem, 8 skippers and 8 boats. For each of our schedules (with the exception of the latter "easy-8" problem) we used 1 day CPU time for the first stage (minimizing boat changes) then a further 2 days to minimize imbalance. We were unable to prove optimality in this time, therefore some of these schedules might be improved. The processor used was an Intel (R) Xeon (R) E5-2660 at 2.20GHz. Note, stages 3 and 4 (renaming and orientation) completed in less than a second.

Flight	Matches
0	(5,2) (4,3) (1,6)
1	(4,2) (6,5) (3,1)
2	(6,4) (2,1) (3,5)
3	(6,2) (5,0) (1,7)
4	(5,1) (0,6) (2,7)
5	(7,5) (2,0) (6,3)
6	(0,7) (4,1) (3,2)
7	(7,4) (0,3)
8	(4,0) (7,3)
9	(5,4) (7,6) (1,0)

Flight	Matches
0	(2,7) (3,0) (5,4)
1	(0,2) (4,3) (7,5)
2	(0,4) (7,6) (5,3)
3	(4,6) (5,0) (3,1)
4	(6,5) (3,2) (1,4)
5	(6,3) (4,7) (2,1)
6	(7,3) (1,6) (4,2)
7	(7,1) (2,5) (6,0)
8	(5,1) (0,7) (6,2)
9	(1,0)

Fig. 6. Schedules for 8 skippers and 6 boats. On the left, the published ISAF schedule (illegal), and on the right is our schedule.

Flight	Matches
0	(0,7) (3,2) (5,4)
1	(0,2) (7,4) (5,3)
2	(4,0) (7,3) (2,5)
3	(3,0) (2,4) (6,5)
4	(4,3) (5,1) (8,6)
5	(3,1) (6,2) (4,8)
6	(6,3) (1,4) (8,2)
7	(4,6) (2,7) (1,8)
8	(7,6) (0,8) (2,1)
9	(8,7) (5,0) (6,1)
10	(8,5) (1,7) (0,6)
11	(7,5) (3,8) (1,0)

Flight	Matches
0	(0,6) (3,2) (5,4) (1,7)
1	(2,0) (6,3) (4,1) (8,7)
2	(6,4) (0,5) (7,2) (3,8)
3	(4,7) (1,8) (0,3) (5,2)
4	(3,1) (7,5) (8,0) (2,6)
5	(3,7) (4,8) (2,1) (6,5)
6	(8,2) (1,6) (5,3) (0,4)
7	(5,1) (7,0) (6,8) (4,3)
8	(8,5) (2,4) (7,6) (1,0)

Fig. 7. Two new schedules. On the left, 9 skippers and 6 boats, and on the right 9 skippers and 8 boats.

8 skippers and 6 boats: We first analyze the ISAF schedule, Figure 6 on the left. Criterion 4 (skippers in last match in a flight cannot be first in next flight) is violated on 3 occasions (skipper 1 in flight 3, skipper 7 in flight 4, skipper 0 in flight 7). Criterion 5 (no more than two consecutive port or starboard assignments) is violated for skipper 6 (flights 1 to 3) and skipper 7 (flights 7 to 9). Criterion 6 (imbalance) is violated for skippers 6 and 8. Criterion 12 (new boats do not sail first) is violated in flight 9 for skipper 5. Finally, the schedule has 8 boat changes. Our schedule, Figure 6 on the right, respects all criteria and has 6 boat changes and a maximum imbalance of 1.

9 skippers: Two new schedules are presented in Figure 7, on the left for 6 boats and on the right for 8 boats. Neither of these schedules appear in the ISAF Umpires' Manual. Both schedules respect all criteria. For 6 boats there are 8 boat changes (no skipper making more than 1 boat change) and a maximum imbalance of 2. For 8 boats there are again 8 boat changes, no skipper with more than 1 boat change, and each skipper 4 times on starboard side and 4 on port side.

10 skippers and 6 boats: Figure 8 shows the ISAF schedule on the left for 10 skippers and 6 boats and on the right, our schedule. The ISAF schedule violates criterion 5 (no more than 2 consecutive port or starboard assignments) ten times (twice for skippers 0, 1 and 6 and once for skippers 3, 7, 8 and 9). Criterion 12 (new boats do not sail first) is violated 7 times (in flight 3 for skippers 6 and 2, flight 5 for skipper 8, flight 6 for skipper 7, flight 8 for skipper 1, flight 9 for skipper 4 and flight 12 for skipper 4 again). There are 22 boat changes, with the minimum changes for a skipper being 1 and the maximum 3. Our schedule, right of Figure 8, satisfies all criteria, has 12 boat changes with the minimum changes for a skipper being 0 and the maximum 2, and a maximum imbalance of 1.

Flight	Matches
0	(8,3) (1,7) (0,9)
1	(7,3) (0,8) (1,9)
2	(0,7) (9,3) (1,8)
3	(6,2) (5,0) (7,4)
4	(5,2) (6,4) (3,0)
5	(8,6) (2,9) (4,1)
6	(7,2) (4,8) (9,6)
7	(2,4) (0,6) (7,5)
8	(6,1) (5,9) (8,2)
9	(9,4) (8,7) (6,5)
10	(9,7) (6,3) (8,5)
11	(7,6) (3,1) (9,8)
12	(4,3) (5,1) (2,0)
13	(3,5) (4,0) (2,1)
14	(5,4) (3,2) (1,0)

Flight	Matches
0	(4,7) (3,2) (0,5)
1	(2,4) (7,0) (3,5)
2	(0,4) (7,3) (5,2)
3	(4,3) (0,2) (5,7)
4	(3,0) (6,5) (2,7)
5	(6,3) (7,1) (8,2)
6	(7,6) (2,1) (9,8)
7	(2,6) (8,7) (4,9)
8	(6,8) (1,4) (7,9)
9	(8,1) (5,4) (9,6)
10	(5,1) (9,2) (4,6)
11	(1,9) (6,0) (8,4)
12	(0,9) (3,8) (6,1)
13	(8,0) (9,5) (1,3)
14	(5,8) (9,3) (1,0)

Fig. 8. Schedules for 10 skippers and 6 boats. On the left, the ISAF published schedule (illegal), and on the right, our schedule.

8 skippers and 8 boats (easy-8): Figure 9 shows the ISAF schedule on the right for 8 skippers and 8 boats and on the right, our schedule. The ISAF schedule violates criterion 4 (last match in a flight cannot be first in next flight) for skipper 5 in flights 1 and 2. Also, criterion 5 (no skipper should have more than 2 consecutive port or starboard assignments) is violated for skippers 0, 1, 3, 4 and 6. Furthermore, skipper 1 appears in the last match of four flights resulting in significant imbalance. Our schedule, on the right, satisfies all criteria, is optimal (maximum imbalance of 1) and took less than 10 seconds in total (all stages) to produce.

Flight	Matches
0	(5,2) (4,3) (1,6) (0,7)
1	(2,4) (0,6) (1,7) (5,3)
2	(0,5) (7,2) (6,3) (4,1)
3	(7,3) (5,1) (4,0) (6,2)
4	(3,0) (4,7) (6,5) (2,1)
5	(6,4) (7,5) (2,0) (3,1)
6	(7,6) (5,4) (3,2) (1,0)

Flight	Matches
0	(4,3) (2,0) (1,5) (7,6)
1	(3,2) (0,4) (6,1) (5,7)
2	(4,1) (7,3) (5,2) (0,6)
3	(1,7) (6,5) (3,0) (2,4)
4	(0,5) (4,6) (7,2) (3,1)
5	(7,0) (2,1) (6,3) (5,4)
6	(6,2) (5,3) (4,7) (1,0)

Fig. 9. Schedules for 8 skippers and 8 boats. On the left, the published ISAF schedule (illegal), and on the right, our schedule.

5 Discussion

It was anticipated that this problem would be easy to model. This naïvety was due to the use of the term "round-robin" for these schedules, leading us to believe that we could use some of the models already in the literature [7, 6, 1]. This naïvety slowly dissolved as we addressed more and more of the ISAF criteria. A number of models we produced were scrapped due to misunderstanding about the actual problem, i.e. a communication problem between the authors. Eventually this was resolved by presenting the majority of the criteria as a deterministic finite automaton. This became our Rosetta Stone, with the surprising benefit that it not only improved the team's communications, it was also a constraint (the *regular* constraint).

The staged approach was taken cautiously. We expected that isolating the orientation of schedules as a post-process might leave us with a hard or insoluble problem. So far, every schedule we have produced has been oriented with very little search, typically taking less than a second.

We used the Choco constraint programming toolkit and one of our goals was to use only toolkit constraints, i.e. we wanted to see how far we could go without implementing our own specialized constraints. We are pleased to say that we did not have to do anything out of the box, although we did feel that we had used just about every constraint there was.

There are many redundant variables in our model. One of our first valid models was essentially that shown in Figure 2, and did not use *modMatch* and *match* variables. Performance was hopeless, struggling to produce a 7 skipper 6 boat schedule in days. The *modMatch* and *match* variables were added. This improved domain filtering, and a flattened *match* array was used as decision variables. That is, the decisions where "when do we schedule this match?". At this point we had not yet produced a 9 skipper 6 boat schedule and we suspected that the combined criteria might exclude such a schedule. A non-constraint model was developed, primarily to attempt to prove that there was no schedule for 9 skippers with 6 boats. This program used a backtracking search with decision variables being positions within flights, i.e. time slots being assigned to matches. At the top of search, the matches in the first flight were set, and symmetry breaking was used to reduce the number of legal second flights. A solution was found in seconds! With this experience we added in the *time* variables, channeled these into the existing model and used these as decision variables, i.e. the question now was "what match will we do now?". We also anchored the matches in the first flight. With these changes we began to produce schedules in acceptable time. The model was then built upon, incrementally adding criteria. This took a surprisingly short amount of time, sometimes minutes of coding to add in a new feature. The model was then enhanced so that it would address optimization rather than just satisfaction, again a trivial programming task.

We have only reported a handful of our schedules, however there are missing schedules i.e. unpublished and not yet produced by us. Examples of these are 10 skippers and 8 boats, 11 skippers with fewer than 10 boats, 12 skippers and 8 boats, 13 skippers and 8 boats and 14 skippers with fewer than 10 boats. None of

Fig. 10. Scene from an imaginary dinghy race.

these schedules have been published by the ISAF, although we expect that they can be produced by selectively violating some of constraints. We have also not yet encoded criterion 7, addressing the situation of 10 or more boats. To do this we will have to modify the DFA used by the *regular* constraint in the first two stages. In addition, we have yet to address the format of *two-group* round-robin schedules as also found in the manual [2].

6 Conclusion

We have produced new and better match race schedules. These schedules can be used by anyone who competes under the criteria published by ISAF. Our schedules can be downloaded as blank schedules and then populated with the names of the actual competing skippers.

So, why did we use constraint programming? The answer is obvious: we are constraint programmers, or to borrow Mark Twain's words "To a man with a hammer, everything looks like a nail". But actually, constraint programming has been a good way to go. From an engineering perspective, it has allowed us to prototype solutions quickly and to build solutions incrementally. There is also an advantage that we might exploit in the future: we can now investigate the effect different criteria have on our ability to produce schedules and how relaxation of those affect optimization criteria. That is, the constraint program might be used to design the next set of criteria for match race schedules, or allow the event organizer to decide which criteria to sacrifice to ensure a faster schedule with fewer boat changes.

References

1. Martin Henz, Tobias Müller, and Sven Thiel. Global constraints for round robin tournament scheduling. *European Journal of Operational Research*, 153(1):92–101, 2004.
2. International Sailing Federation. ISAF International Umpires' and Match Racing Manual, 2012.
3. Michele Lombardi and Michela Milano. Optimal methods for resource allocation and scheduling: a cross-disciplinary survey. *Constraints*, 17(1):51–85, 2012.
4. Gilles Pesant. A regular language membership constraint for finite sequences of variables. In *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, pages 482–495, 2004.
5. Claude-Guy Quimper, Peter van Beek, Alejandro López-Ortiz, Alexander Golynski, and Sayyed Bashir Sadjad. An efficient bounds consistency algorithm for the global cardinality constraint. In Francesca Rossi, editor, *Principles and Practice of Constraint Programming CP 2003*, volume 2833 of *Lecture Notes in Computer Science*, pages 600–614. Springer Berlin Heidelberg, 2003.
6. Rasmus V. Rasmussen and Michael A. Trick. Round robin scheduling - a survey. *European Journal of Operational Research*, 188(3):617–636, 2008.
7. Michael A. Trick. Integer and constraint programming approaches for round-robin tournament scheduling. In *Practice and Theory of Automated Timetabling IV, 4th International Conference, PATAT 2002, Gent, Belgium, August 21-23, 2002, Selected Revised Papers*, pages 63–77, 2002.